Hyperopt-Sklearn

Automatic Hyperparameter Configuration for Scikit-Learn

Brent Komer, James Bergstra, Chris Eliasmith {bjkomer, james.bergstra, celiasmith}@uwaterloo.ca
Centre for Theoretical Neuroscience, University of Waterloo <http://ctn.uwaterloo.ca>

Abstract

Hyperopt-sklearn is a new software project that provides automatic algorithm configuration of the Scikit-learn machine learning library. Following Auto-Weka, we take the view that the choice of classifier and even the choice of pre-processing module can be taken together to represent a single large hyperparameter optimization problem. We use Hyperopt to define a search space that encompasses many standard components (e.g. SVM, RF, KNN, PCA, TFIDF) and common patterns of composing them together. We demonstrate, using search algorithms in Hyperopt and standard benchmarking data sets (MNIST, 20-Newsgroups, Convex Shapes), that searching this space is practical and effective. In particular, we improve on best-known scores for the model space for both MNIST and Convex Shapes.

Experiments

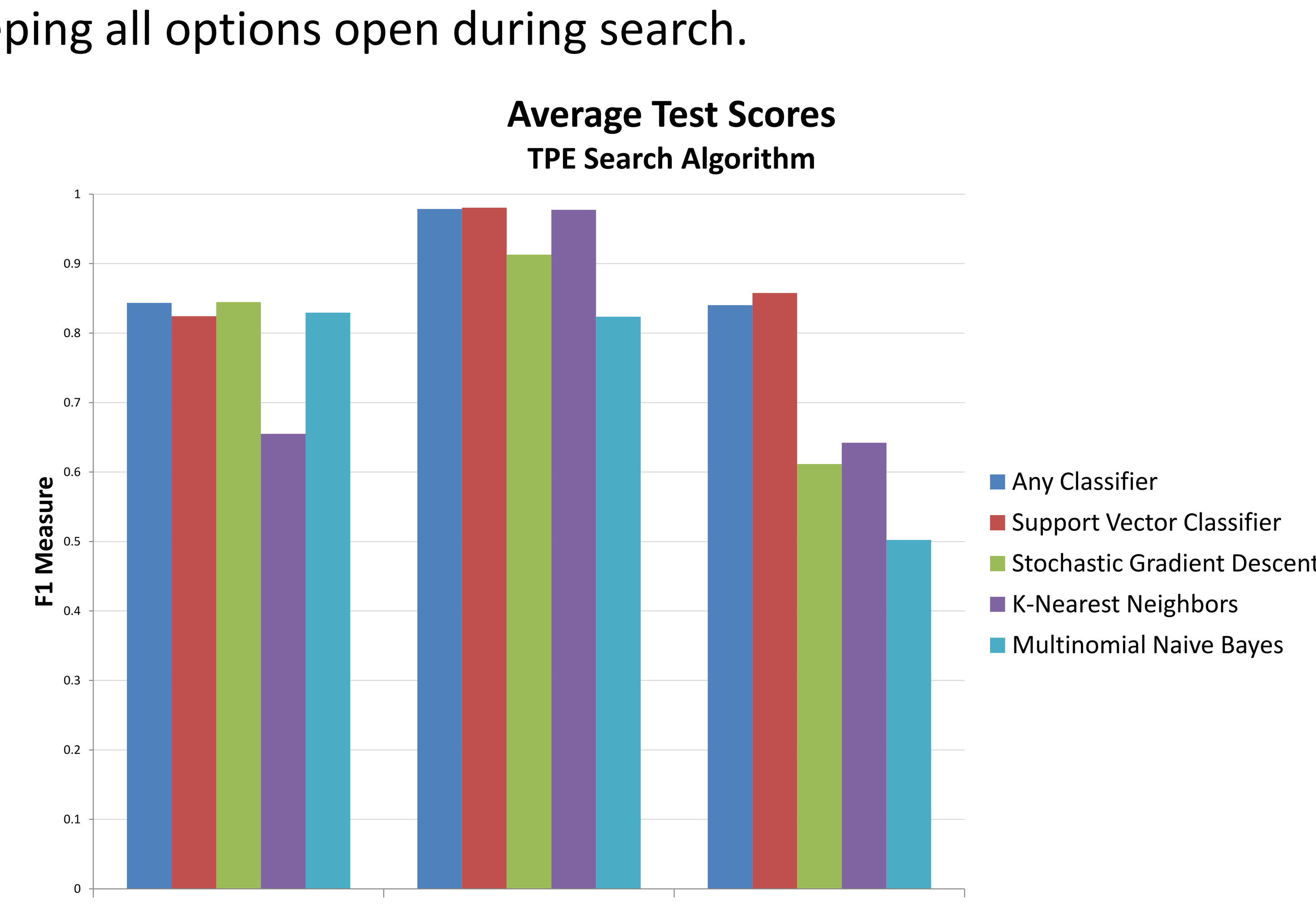
Three data sets were used to conduct experiments on the effectiveness of hyperopt-sklearn.

MNIST Digits: A set of 28x28 greyscale images of hand drawn digits (60,000 images in the training set, 10,000 in the test set)

20-Newsgroups: A corpus of newsgroup messages that can be classified into 20 different categories (11314 articles in training set, 7532 articles in test set, used all 20 categories)

Convex Shapes: binary classification task of distinguishing pictures of convex white-coloured regions in 32x32 black-and-white images (8,000 images in training set, 50,000 in test set)

Optimization runs were performed on both the entire search space as well as subspaces corresponding to specific classifier types. Most experiments were run for 300 function evaluations of the parameter space. We used three optimization algorithms available in Hyperopt: random search, annealing, and TPE. The performance of the model found from searching the entire space was not statistically inferior to the best model pulled from each classifier subspace; there was no penalty for keeping all options open during search.

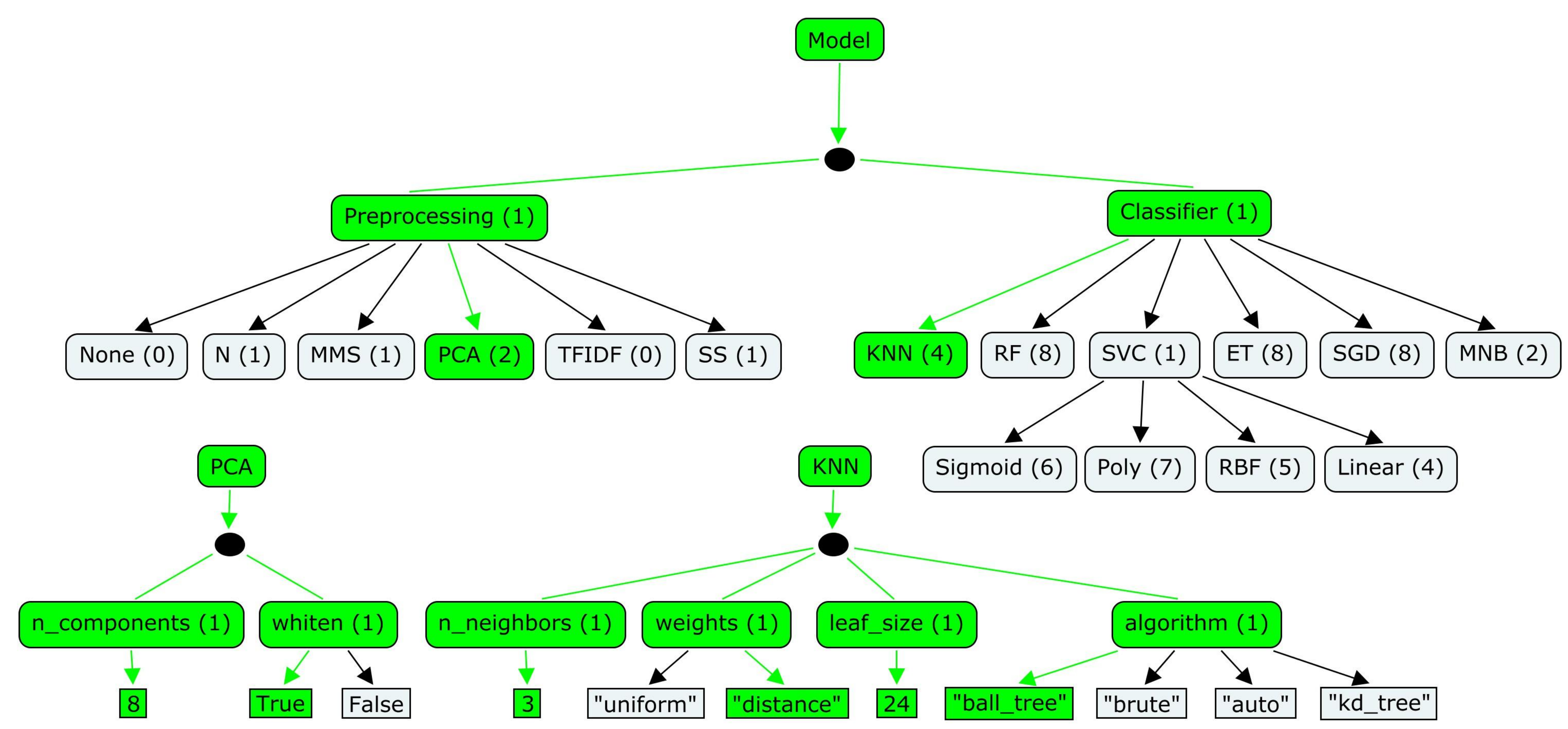


Project Websites

Hyperopt: http://hyperopt.github.io/hyperopt
Scikit-Learn: http://scikit-learn.org/
Hyperopt-Sklearn: http://hyperopt.github.io/hyperopt-sklearn/
AutoWEKA: http://www.cs.ubc.ca/labs/beta/Projects/autoweka/

Search Space

Highlighted in green is one possible configuration of parameters in this space, which in this case is a PCA and K-Nearest Neighbors model. The number of active hyperparameters in a model is the sum of parenthetical numbers in the selected boxes.




```
RF:    Random Forest Classifier          PCA:   Principal Component Analysis
KNN:   K-Nearest Neighbors               TFIDF: Term Frequency - Inv Doc Freq
SVC:   Support Vector Classifier         SS:    Standard Scaler
ET:    Extra Trees Classifier            MMS:   Min Max Scaler
SGD:   Stochastic Gradient Descent       N:     Normalizer
MNB:   Multinomial Naive Bayes
```

Comparison to Previous Work

MNIST		20 Newsgroups		Convex Shapes	
Approach	**Accuracy**	**Approach**	**F-Score**	**Approach**	**Accuracy**
Committee of convnets	99.8%	CFC	0.928	**Hyperopt-sklearn**	**88.7%**
Hyperopt-sklearn	**98.7%**	**Hyperopt-sklearn**	**0.856**	Hp-dbnet	84.6%
libSVM grid search	98.6%	SVMTorch	0.848	Dbn-3	81.4%
Boosted trees	98.5%	LibSVM	0.843		

In the 20 Newsgroups dataset, the score reported for hyperopt-sklearn is the weighted-average F1 score provided by sklearn. The other approaches shown here use the macro-average F1 score.

Example Usage

```
from hpsklearn import hyperopt_estimator, any_sparse_classifier, tfidf
from sklearn.datasets import fetch_20newsgroups
from hyperopt import tpe

# Download the data and split into training and test sets
train = fetch_20newsgroups( subset='train' )
test = fetch_20newsgroups( subset='test' )
X_train = train.data
y_train = train.target
X_test = test.data
y_test = test.target

# Configure the search space
estim = hyperopt_estimator( classifier=any_sparse_classifier('clf'),
                            preprocessing=[tfidf('tfidf')],
                            algo=tpe.suggest, trial_timeout=300)

# Search the space by evaluating points and recording the validation score
estim.fit( X_train, y_train )

# Report the parameters used in the best model found by the fit method
print( estim.best_model() )

# Report the accuracy of the model on the test set
print( estim.score( X_test, y_test ) )
```